#2

Docket No. 1538.1009/JDH

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:	)	
	)	
Yutaka YAMANAKA, et al.	)	
	)	Group Art Unit: Unassigned
Serial No.: TO BE ASSIGNED	)	
	)	Examiner: Unassigned
Filed: January 31, 2001	)	
	)	
For: COMPILER FRO PARALLEL COMPUTER	)	

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. §1.55

Assistant Commissioner for Patents
Washington, D.C. 2023l

Sir:

In accordance with the provisions of 37 C.F.R. §1.55, the applicant submits herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2000-305608, filed October 5, 2000.

It is respectfully requested that the applicant be given the benefit of the foreign filing date as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. §119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: January 31, 2001

By: _____

James D. Halsey, Jr.
Registration No. 22,729

700 11th Street, N.W., Ste. 500
Washington, D.C. 20001
(202) 434-1500

日　本　国　特　許　庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年１０月　５日

出願番号
Application Number: 特願２０００－３０５６０８

出願人
Applicant (s): 富士通株式会社

２０００年１２月１５日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２０００－３１０４７２０

【書類名】　特許願
【整理番号】　0051274
【提出日】　平成12年10月 5日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 09/45
【発明の名称】　コンパイラ
【請求項の数】　5
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　山中　豊
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　石川　貴洋
【特許出願人】
【識別番号】　000005223
【氏名又は名称】　富士通株式会社
【代理人】
【識別番号】　100103528
【弁理士】
【氏名又は名称】　原田　一男
【電話番号】　045-290-2761
【手数料の表示】
【予納台帳番号】　076762
【納付金額】　21,000円
【提出物件の目録】
【物件名】　明細書　1
【物件名】　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9909129

【プルーフの要否】　　要

【書類名】　　明細書

【発明の名称】　　コンパイラ

【特許請求の範囲】

【請求項１】

ソース・プログラムをコンパイルするコンパイラを格納した記録媒体であって、

前記コンパイラは、コンピュータに、

前記ソース・プログラム中の並列化指示文を検出するステップと、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップと、

を実行させるためのコンパイラである、記録媒体。

【請求項２】

前記並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から前記並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップ、

をさらに実行させるためのコンパイラであることを特徴とする請求項１記載の記録媒体。

【請求項３】

各前記処理コードに対応する１又は複数の処理情報を格納する処理テーブルを用いて、前記並列化指示文に対するフロントエンド内中間言内の前記処理コードから現在の処理内容に対応する前記処理情報を取得するステップと、

を実行させるためのコンパイラであることを特徴とする請求項１記載の記録媒体。

【請求項４】

前記現在の処理内容が、型解析処理、構文解析処理、意味解析処理、又はコンパイラ内中間言生成であることを特徴とする請求項３記載の記録媒体。

【請求項５】

ソース・プログラムをコンパイルするコンパイル装置であって、

前記ソース・プログラム中の並列化指示文を検出する手段と、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成する手段と、

を有するコンパイル装置。

【発明の詳細な説明】

【０００１】

【発明が属する技術分野】

本発明は、コンパイラに関し、より詳しくは並列コンピュータに対するソース・プログラムをコンパイルするコンパイラに関する。

【０００２】

【従来の技術】

近年、複数のＣＰＵ（Central Processing Unit）を有するコンピュータは、ＣＰＵ価格の下落等の理由で普及してきている。よって、例えばＯｐｅｎＭＰといった、共有メモリ型の並列コンピュータ向けプログラミングにおけるＡＰＩ（Application Program Interface）の使用も広がってきている。このような状況の下、並列コンピュータのためのコンパイラの重要性も上がってきている。

【０００３】

【発明が解決しようとする課題】

しかし、従来では並列化指示文がソース・プログラムにおいて指定されていた時に、コンパイラでは並列化指示文内の構造情報を保持することなく処理していた。また、並列化指示文と当該並列化指示文が有効となる文（式を含む）との関係を示す情報を保持することなく処理していた。このため、コンパイラにおける処理が複雑になり、且つ速度の向上を阻害していた。

【０００４】

よって本発明の目的は、新規のデータ構造を導入して高速処理可能な、並列コンピュータのためのコンパイラを提供することである。

【０００５】

【課題を解決するための手段】

本発明に係る、並列化指示文を含むソース・プログラムをコンパイルするコンパイラは、コンピュータに、当該ソース・プログラム中の並列化指示文を検出するステップと、

【０００６】

並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップとを実行させる。

【０００７】

これにより構文解析、意味解析などの解析処理で階層構造を有するフロントエンド内中間言を使用することができるようになり、その都度階層構造を確認しながら解析処理を行っていた従来よりも処理が高速化される。

【０００８】

また、並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から当該並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップをさらに実行させるようにする。これにより、並列化指示文とそれが有効となる文との関係が明確となる。使用する記憶領域を削減する効果もある。

【０００９】

なお、上で述べたようなコンパイラが通常のコンピュータで実行されれば当該コンピュータはコンパイル装置となる。また、コンパイラは、例えばフロッピーディスク、ＣＤ－ＲＯＭ、光磁気ディスク、半導体メモリ、ハードディスク等の記憶媒体又は記憶装置に格納される。また、コンパイラの処理途中の中間的なデータは、コンピュータのメインメモリなどの記憶装置に格納される。

【００１０】

【発明の実施の形態】

本発明の一実施の形態に係る機能ブロック図を図１に示す。コンピュータ１は

、ＯｐｅｎＭＰなどのＡＰＩに従った並列化指示文を含むソースプログラムのファイル１０をコンパイルし、生成したオブジェクトコードのファイル３０を出力するコンパイラ２０を実行する。コンパイラ２０には、プログラミング言語に従って書かれたソースプログラムに含まれる各字句を解析する字句解析部２１と、ソースプログラムの構造を解析する構文解析部２２と、ソースプログラムの意味内容を解析する意味解析部２３と、中間言変換部２４とが含まれる。この字句解析部２１と構文解析部２２と意味解析部２３と中間言変換部２４はフロントエンド部４０と呼ばれ、フロントエンド部４０内で用いられる中間言をフロントエンド内中間言と呼ぶ。中間言変換部２４は、フロントエンド内中間言を最適化処理などで用いるコンパイラ内中間言（単に「中間言」と呼ばれることもある。）に変換する。またコンパイラ２０には、最適化処理を行う最適化処理部２５と、最適化処理などの結果を用いてオブジェクトコードを生成するコード生成部２６とがさらに含まれる。

【００１１】

本実施の形態では構文解析部２２に以下で説明する処理を追加することにより、ＯｐｅｎＭＰなどに従って記述された並列化指示文をリスト構造化して、フロントエンド内中間言を生成する。リスト構造は階層構造の一種である。

【００１２】

例えば、

```
#pragma omp parallel private(a,b) lastprivate(c)
```

上の並列化指示文が有効となる文

といった文がソースプログラムに含まれている場合には、図２に示すようなデータ構造を生成する。なお、「#pragma omp」はＯｐｅｎＭＰのＡＰＩであることを示すものである。また、上の例ではparallelが指示子（directive）、private及びlastprivateが節（clause）、ａ，ｂ及びｃが並びと呼ばれている。

【００１３】

図２では、並列化指示文が有効となる文のコード格納域２０１と、その文についての各種情報格納域２０３とが含まれる。例えば、並列化指示文がＦＯＲ文で

あれば、コード格納域２０１にはＦＯＲ文であることを示すコードが格納され、各種情報格納域２０３にはループの情報等が格納される。なお、ここまでは従来と変わらない。この領域をa_statementと呼ぶ。本実施の形態では、並列化指示文の指示子（上の例ではparallel）のアドレス格納域２０５が、並列化指示文が有効となる文のフロントエンド内中間言において付加されている。これにより、以下に示すリスト構造へのアクセスが容易になり、並列化指示文と当該並列化指示文が有効となる文との関係が明確化される。また、並列化指示文の存在のみを示すフロントエンド内中間言が不要になる。

【００１４】

指示子のアドレス格納域２０５に格納されているアドレス（＊１）が、指示子parallelに関連する節へのリンク情報等が格納される情報域２０７の先頭となる。また、情報域２０７に対応して、parallelに対応する処理コードが格納される情報域２０９も設けられている。また、節priveteに関連する並びへのリンク情報及び節lastprivateへのリンク情報等が格納される情報域２１１と、privateに対応する処理コードが格納される情報域２１３とが対応して設けられている。さらに、節lastprivateに関連する並びへのリンク情報等が格納される情報域２１５と、lastprivateに対応する処理コードが格納される情報域２１７とが対応して設けられる。

【００１５】

節privateに関連する並びａに関連する他の並びへのリンク情報等が格納される情報域２１９と、ａに対応する処理コードが格納される情報域２２１とが対応して設けられる。また、節privateに関連する並びｂに関する情報が格納される情報域２２３（上の例では空）と、ｂに対応する処理コードが格納される情報域２２５とが対応して設けられる。また、節lastprivateに関連する並びｃに関する情報が格納される情報域２２７（上の例では空）と、ｃに対応する処理コードが格納される情報域２２９とが対応して設けられる。なお、処理コードは、指示子、節、並びの種類を区別することも可能な一意のコード（数値）である。例えばparallelは０ｘ０１、privateは０ｘ０２である。

【００１６】

このようなデータ構造を生成するために、従来の処理フローに加えて実施される処理フローを図３を用いて説明する。最初に並列化指示文か否かを判断する（ステップＳ１）。並列化指示文でなければ元の処理に戻る。並列化指示文である場合には、指示子があるか判断する（ステップＳ３）。並列化指示文であって指示子がない場合には、エラーであるから元の処理に戻る。もし指示子がある場合には、指示子の処理コードをリスト構造に設定する（ステップＳ５）。これがリスト構造の先頭領域となる。なお図２の例ではparallelの処理コードを設定する。そして、並列化指示文が有効となる文のフロントエンド中間言に、リスト構造の起点を登録する（ステップＳ７）。図２の例では指示子のアドレス格納域２０５に、指示子parallelの情報域２０７のアドレスを格納する。

【００１７】

以下の説明では、情報域２０７及び２０９、情報域２１１及び２１３、情報域２１５及び２１７、情報域２１９及び２２１、情報域２２３及び２２５、情報域２２７及び２２９をそれぞれ１つの情報域として説明する場合もある。

【００１８】

次に未処理の節が存在するか判断する（ステップＳ９）。未処理の節が存在しない場合には、元の処理に戻る。もし未処理の節が存在している場合には、節の処理コードをリスト構造に設定する（ステップＳ１１）。なお、節の処理コードをリスト構造に設定した場合には、最初は当該節に関連する指示子の情報域に当該節の情報域のアドレスを格納する。また、節が２つ以上ある場合において、２番目以降の節については前の節の情報域に当該節の情報域のアドレスを格納する。図２の例では、指示子parallelに関連する節はprivateとlastprivateの２つあり、節privateの情報域２１１のアドレスは指示子parallelの情報域２０７に格納され、節lastprivateの情報域２１５のアドレスは節privateの情報域２１１に格納される。

【００１９】

そして未処理の並びが存在するか判断する（ステップＳ１３）。もし未処理の並びがなければステップＳ９に戻る。一方、未処理の並びが存在している場合には、並びの処理コードをリスト構造に設定する（ステップＳ１５）。なお、並び

の処理コードをリスト構造に設定した場合には、最初は当該並びに関連する節の情報域に当該並びの情報域のアドレスを格納する。また、並びが２つ以上ある場合において、２番目以降の並びについては前の並びの情報域に当該並びの情報域のアドレスを格納する。図２の例では、節privateに関連する並びはａとｂの２つあり、並びａの情報域２１９のアドレスは節privateの情報域２１１に格納され、並びｂの情報域２２３のアドレスは並びａの情報域２１９に格納される。なお、節lastprivateに関連する並びはｃだけであり、並びｃの情報域２２７のアドレスは節lastprivateの情報域２１５に格納される。

【００２０】

ステップＳ１３及びステップＳ１５は未処理の並びが無くなるまで繰り返される。図３に示すような処理により図２に示されるようなデータ構造が生成される。

【００２１】

図２に示されるようなデータ構造は、図４に示すような処理テーブルと合わせて使用される。図４の処理テーブルでは、列４０１が処理１についての情報を格納しており、列４０３が処理２についての情報を格納している。また、行４０５がparallelの処理コードに対応する、処理１、処理２．．．についての情報を格納しており、行４０７がprivateの処理コードに対応する、処理１、処理２．．．についての情報を格納しており、行４０９がvariableの処理コードに対応する処理１、処理２．．．についての情報を格納している。

【００２２】

処理１、処理２といった処理は、例えば構文解析の情報取得又は処理関数実行や、意味解析の情報取得又は処理関数実行、型チェックの情報取得又は処理関数実行、コンパイラ内中間言の取得といった、構文解析部２２、意味解析部２３及び中間言変換部２４において実行される各種処理である。

【００２３】

例えば、処理１を型チェックのための情報取得とし、例えば図４のａ１をｏｎ、図４のａ２をｏｆｆとする。そうすると、型チェックのための情報取得を行う際には、parallelの処理コードに対してはｏｎという情報が、privateの処理コ

ードに対してはｏｆｆという情報が取得できる。同様に、処理２を意味解析用関数アドレスとし、図４のｂ１をparallel用の意味解析用関数アドレスとし、ｂ２をprivate用の意味解析用関数アドレスとする。そうすると意味解析処理を行う際には、parallelの処理コードに対してparallel用の意味解析用関数アドレスを取得することができ、その関数を実行することができる。また、privateの処理コードに対してはprivate用の意味解析用関数アドレスを取得することができ、その関数を実行することができる。

【００２４】

本実施の形態では、処理テーブル内の情報を取得する場合には、例えば以下のような式を用いる。

テーブル情報＝処理テーブル［処理コード］．処理ｘ

ここで処理ｘとは、処理１、処理２．．．を表す。このように処理テーブルから取得した「テーブル情報」を用いて処理を行うことができる。例えば、処理１がコンパイラ内中間言の取得であって、ａ１がparallel（処理コード0x01）のコンパイラ内中間言であるparallel_xであり、ａ２がprivate（処理コード0x02）のコンパイラ内中間言であるprivate_xとすると、

parallelコンパイラ内中間言＝処理テーブル［０ｘ０１］．処理１

でparallelコンパイラ内中間言にparallel_xが入る。

privateコンパイラ内中間言＝処理テーブル［０ｘ０２］．処理１

でprivateコンパイラ内中間言にprivate_xが入る。

【００２５】

一例として中間言変換部２４における、処理テーブル及び処理コードを用いたフロントエンド内中間言からコンパイラ内中間言への変換処理（本実施の形態における変更分）を図５を用いて説明する。

【００２６】

最初にリスト構造を有する文のフロントエンド内中間言であるか否かを判断する（ステップＳ２１）。本実施の形態の場合、指示子のアドレス格納域が存在しているか検査すれば良い。もし、リスト構造を有する文のフロントエンド内中間言でなければ、元の処理に戻る。リスト構造を有する文のフロントエンド内中間

言であれば、並列化指示文のコンパイラ内中間言を生成する（ステップＳ２３）。並列化指示文のコンパイラ内中間言は、従来と同じである。

【００２７】

そして指示子が存在しているか判断する（ステップＳ２５）。指示子が存在しているかは、リスト構造において最初の処理コードを調べればよい。処理コードは、指示子、節、並びの別をも示している。もし、指示子が存在していなければ元の処理に戻る。指示子が存在していれば、指示子のコンパイラ内中間言を処理テーブルを用いて設定する（ステップＳ２７）。ここでは、指示子コンパイラ内中間言＝処理テーブル［指示子の処理コード］．処理１（処理１がコンパイラ内中間言の取得である場合）という処理を行う。

【００２８】

次に未処理の節が存在しているか判断する（ステップＳ２９）。ステップＳ２９の最初の実施の場合には、リスト構造内の指示子の情報域に格納されている他の情報域のアドレスが存在しているか否かで判断する。ステップＳ２９の実施が２度目以降の場合には、前の節の情報域内に、並びの情報域以外の情報域のアドレスが格納されているかで判断する。もし、未処理の節が存在していない場合には、元の処理に戻る。未処理の節が存在している場合には、節のコンパイラ内中間言を処理テーブルを用いて設定する（ステップＳ３１）。節の情報域のアドレスにアクセスし、その節の処理コードを取得する。そして、節コンパイラ内中間言＝処理テーブル［節の処理コード］．処理１という処理を行う。

【００２９】

次に、未処理の並びが存在しているか判断する（ステップＳ３３）。ステップＳ３３の最初の実施の場合には、リスト構造内の節の情報域内に、他の情報域のアドレスが存在するか確認し、もし存在する場合には当該他の情報域のアドレスにアクセスしてその情報域に格納された処理コードが並びの処理コードか否かで判断する。また、ステップＳ３３の２度目以降の実施の場合には、現在の並びの情報域に他の情報域へのアドレスが格納されているかで判断する。もし、未処理の並びが存在していない場合には、ステップＳ２９に戻る。一方、未処理の並びが存在している場合には、並びのコンパイラ内中間言を処理テーブルを用いて設

定する（ステップＳ３５）。並びの情報域のアドレスにアクセスし、並びの処理コードを取得する。そして、並びコンパイラ内中間言＝処理テーブル［並びの処理コード］．処理１という処理を行う。

【００３０】

以上のような処理を行うことにより、フロントエンド内中間言からコンパイラ内中間言を生成することができるようになる。指示子、節及び並びの処理コードを含むリスト構造と処理テーブルを保持しているため、無駄な解析処理を行わずに済むので、簡単且つ高速にコンパイラ内中間言を処理テーブルから取得することができる。

【００３１】

なお、ステップＳ２５以降の処理は、他の処理、例えば、意味解析用情報の取得や、意味解析処理の実施（意味解析用関数関数アドレスの取得と実行）等の場合でも、ステップＳ２７、ステップＳ３１及びステップＳ３３を「テーブル情報＝処理テーブル［処理コード］．処理ｘ」といった形の処理に変更すれば、他の処理に適用することができる。その場合には、無駄な解析処理を行わずに済むので、簡単且つ高速に所望の処理を実施することができるようになる。また、新たな処理が必要になった場合には、処理テーブルの列の数を増やせば簡単に対処することができるようになる。

【００３２】

以上本発明の一実施の形態を説明したが、様々な変形が可能である。例えば図１において最適化処理部２５の後にコード生成部２６が設けられているが、最適化処理の他に他の処理（例えばレジスタ割付処理）を実施した後にコード生成処理を行うような構成であってもよい。また、リスト構造は図２のような構造でなくとも、並列化指示文の構造が表された階層的なデータ構造であっても良い。また、図３及び図５の処理フローはＯｐｅｎＭＰにおける並列化指示文を前提とした処理フローとなっているので、他のルールに従った並列化指示文の場合にはその並列化指示文の構造に従って変更する必要がある。上でも述べたが図４の処理テーブルは、列を追加すれば様々な処理に対処できるようになり、行を増やせば取扱い処理コードの数が増える。

【００３３】

また、図１のコンピュータ１がネットワークに接続されており、ソースプログラム・ファイル１０が他のコンピュータから送信されてきたり、オブジェクトコード・ファイル３０が他のコンピュータへ送信されるような場合もある。他のコンピュータは並列コンピュータである場合もある。

【００３４】

さらに上で述べたコンパイラは通常のコンピュータにおいて実行されるプログラムの形態で実施され、その場合、コンパイラは、例えばフロッピーディスク、ＣＤ－ＲＯＭ、光磁気ディスク、半導体メモリ、ハードディスク等の記憶媒体又は記憶装置に格納される。プログラムであるコンパイラは、ネットワークを介して配布される場合もある。

【００３５】

（付記１）

ソース・プログラムをコンパイルするコンパイラを格納した記録媒体であって、

前記コンパイラは、コンピュータに、

前記ソース・プログラム中の並列化指示文を検出するステップと、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップと、

を実行させるためのコンパイラである、記録媒体。

【００３６】

（付記２）

前記並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から前記並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップ、

をさらに実行させるためのコンパイラであることを特徴とする付記１記載の記録媒体。

【００３７】

（付記３）

各前記処理コードに対応する１又は複数の処理情報を格納する処理テーブルを用いて、前記並列化指示文に対するフロントエンド内中間言内の前記処理コードから現在の処理内容に対応する前記処理情報を取得するステップと、

を実行させるためのコンパイラであることを特徴とする付記１記載の記録媒体。

【００３８】

（付記４）

前記現在の処理内容が、型解析、構文解析、意味解析、又はコンパイラ内中間言生成であることを特徴とする付記３記載の記録媒体。

【００３９】

（付記５）

前記階層構造が、リスト構造であることを特徴とする付記１記載の記録媒体。

【００４０】

（付記６）

前記並列化指示文の部分には、指示子、節及び並びが含まれ、

前記指示子の処理コードの下位に前記節の処理コード、前記節の処理コードの下位に並びの処理コードがリンク付けされる

ことを特徴とする付記１記載の記録媒体。

【００４１】

（付記７）

ソース・プログラムをコンパイルするコンパイル装置であって、

前記ソース・プログラム中の並列化指示文を検出する手段と、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成する手段と、

を有するコンパイル装置。

【００４２】

【発明の効果】

以上のように、新規のデータ構造を導入して高速処理可能な、並列コンピュータのためのコンパイラを提供することができた。

【図面の簡単な説明】

【図１】

本発明の一実施の形態に係るコンパイラを実行するコンピュータの機能ブロック図である。

【図２】

本発明の一実施の形態に係るフロントエンド内中間言の一例を示す図である。

【図３】

図２のデータ構造を生成するための処理フローを示す図である。

【図４】

処理テーブルの一例を示す図である。

【図５】

フロントエンド内中間言からコンパイラ内中間言を生成するための処理フローを表す図である。

【符号の説明】

１　コンピュータ　　１０　ソースプログラム・ファイル

２０　コンパイラ　　２１　字句解析部　　２２　構文解析部

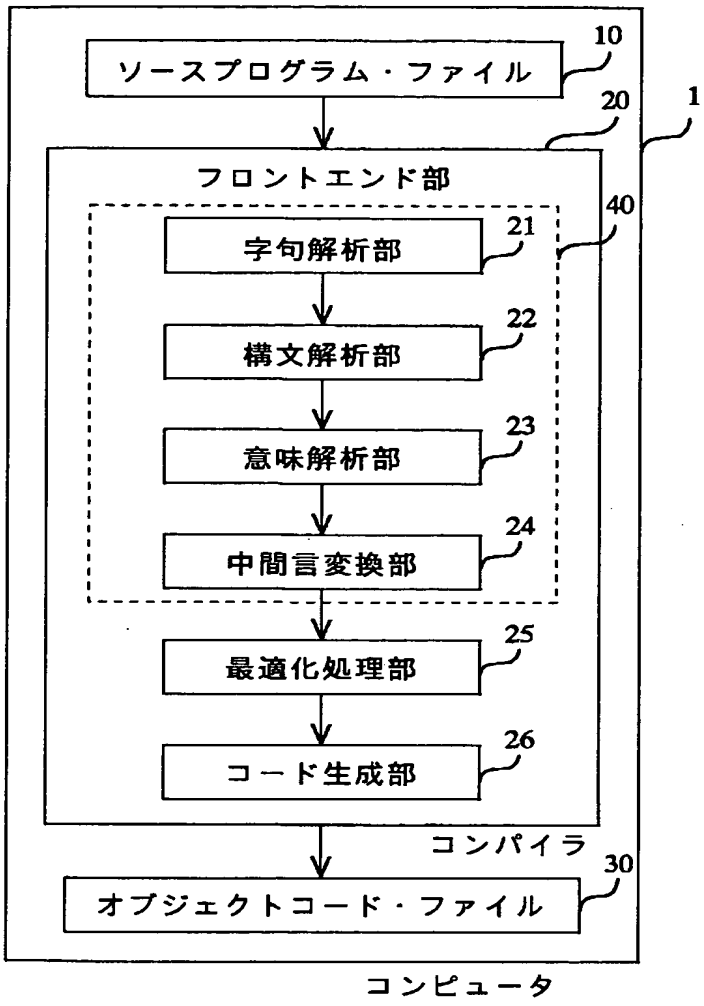
２３　意味解析部　　２４　中間言変換部　　２５　最適化処理部

２６　コード生成部　　３０　オブジェクトコード・ファイル

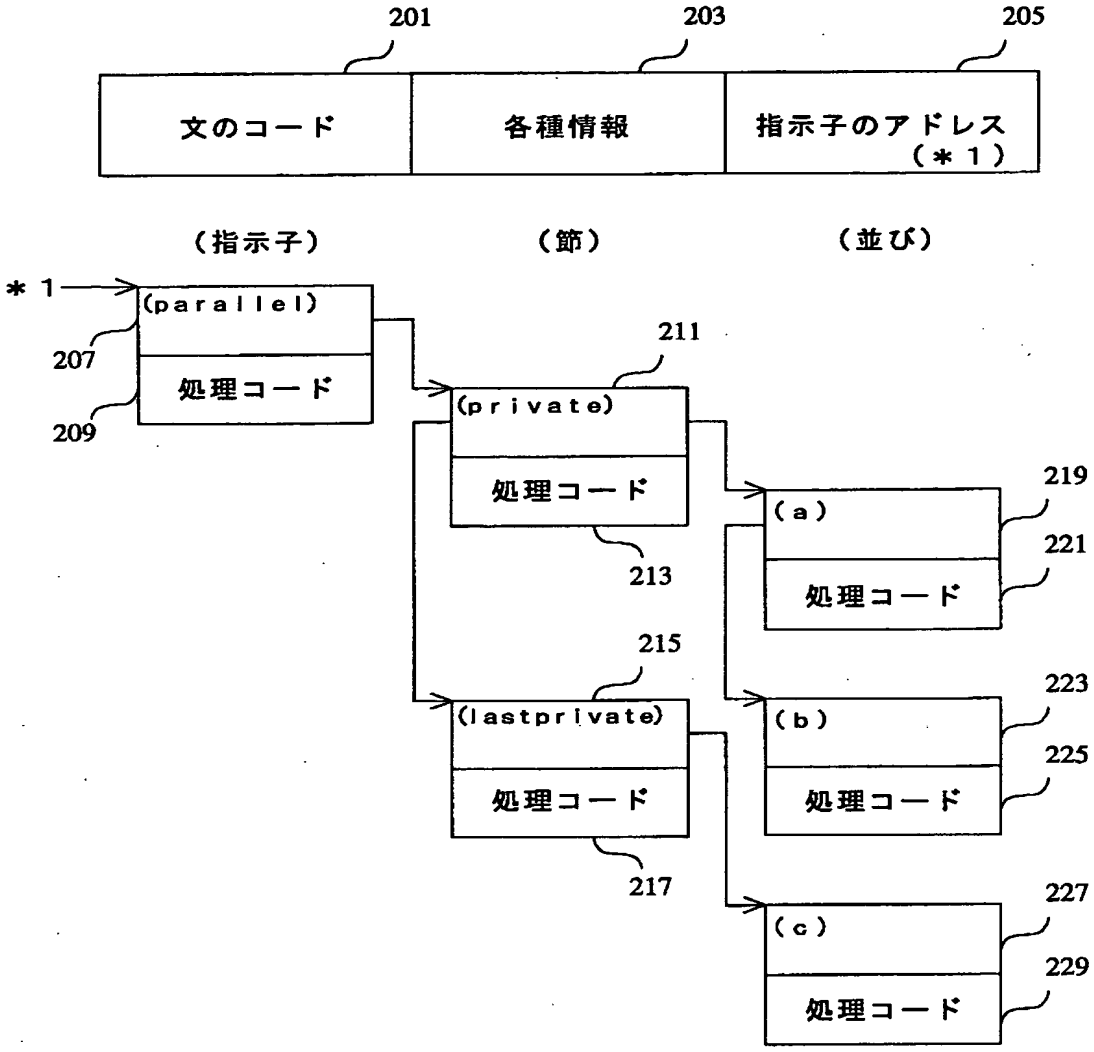
４０　フロントエンド部

【書類名】　　図面

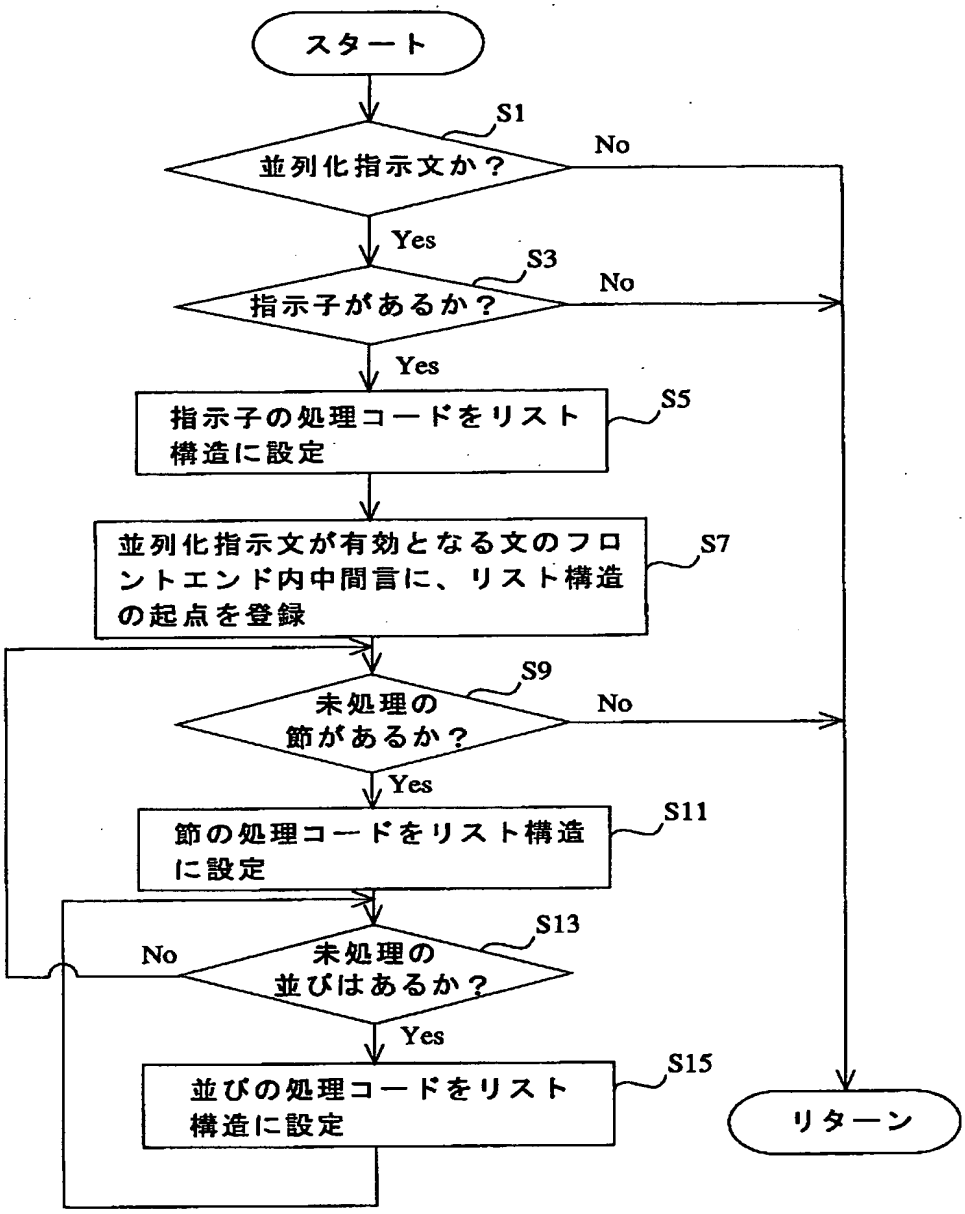
【図１】


ソースプログラム・ファイル
10
1
20
フロントエンド部
40
字句解析部
21
構文解析部
22
意味解析部
23
中間言変換部
24
最適化処理部
25
コード生成部
26
コンパイラ
オブジェクトコード・ファイル
30
コンピュータ

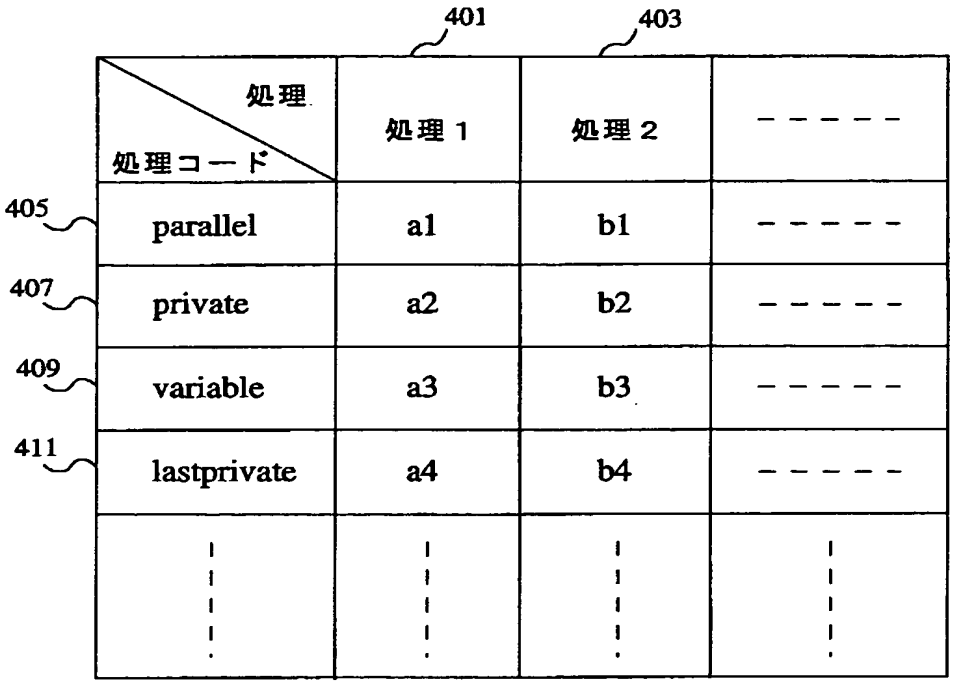
【図２】


201
203
205
文のコード
各種情報
指示子のアドレス
（＊１）
（指示子）
（節）
（並び）
＊１
(parallel)
207
処理コード
209
211
(private)
処理コード
213
219
(a)
221
処理コード
215
(lastprivate)
処理コード
217
223
(b)
225
処理コード
227
(c)
229
処理コード

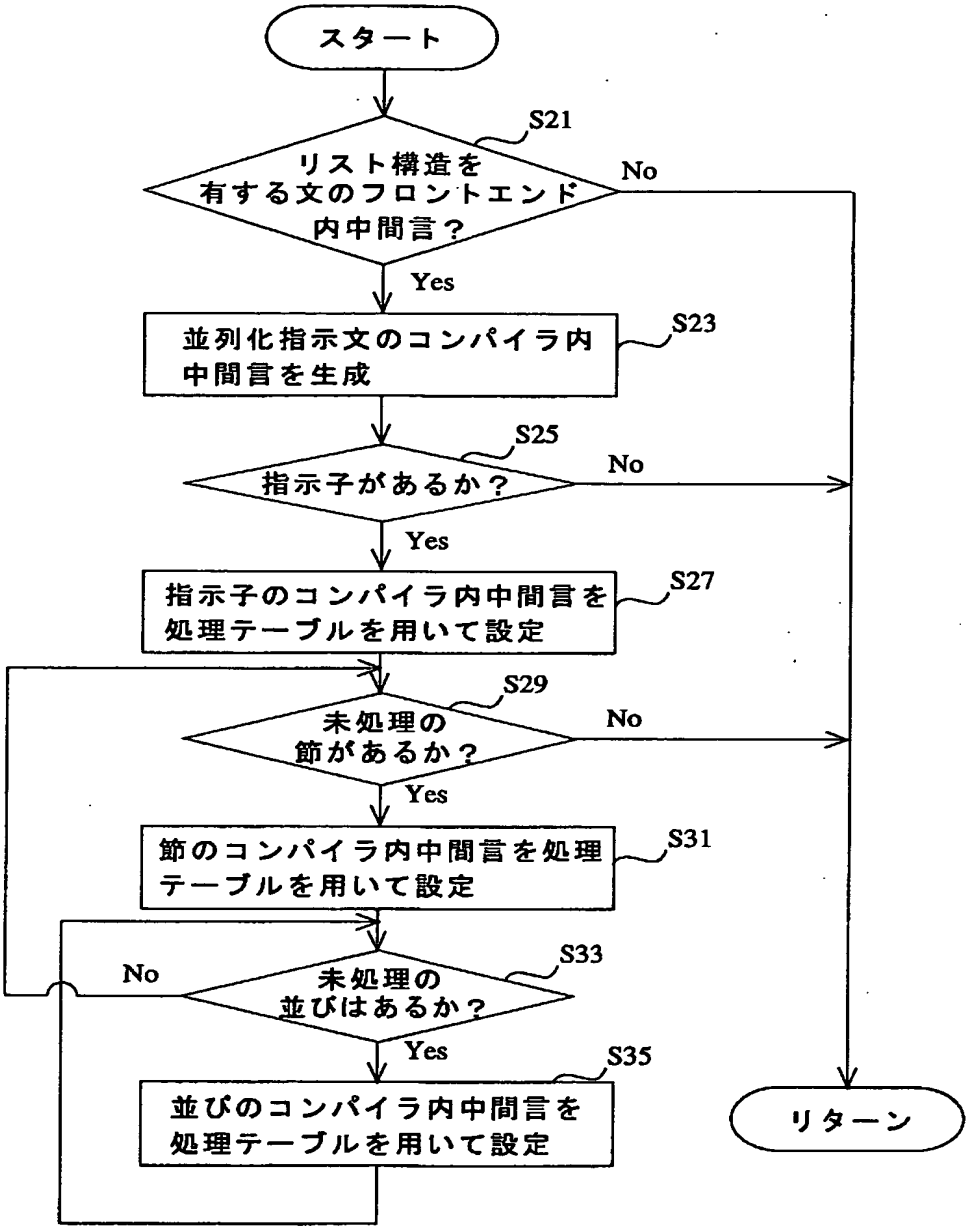
【図３】


スタート
S1
並列化指示文か？
No
Yes
S3
指示子があるか？
No
Yes
指示子の処理コードをリスト構造に設定
S5
並列化指示文が有効となる文のフロントエンド内中間言に、リスト構造の起点を登録
S7
S9
未処理の節があるか？
No
Yes
節の処理コードをリスト構造に設定
S11
S13
No
未処理の並びはあるか？
Yes
並びの処理コードをリスト構造に設定
S15
リターン

【図４】


401
403
処理
処理コード
処理１
処理２
－－－－－
405
parallel
a1
b1
－－－－－
407
private
a2
b2
－－－－－
409
variable
a3
b3
－－－－－
411
lastprivate
a4
b4
－－－－－

【図５】


スタート
S21
リスト構造を有する文のフロントエンド内中間言？
No
Yes
並列化指示文のコンパイラ内中間言を生成
S23
S25
指示子があるか？
No
Yes
指示子のコンパイラ内中間言を処理テーブルを用いて設定
S27
S29
未処理の節があるか？
No
Yes
節のコンパイラ内中間言を処理テーブルを用いて設定
S31
S33
No
未処理の並びはあるか？
Yes
S35
並びのコンパイラ内中間言を処理テーブルを用いて設定
リターン

【書類名】　　要約書

【要約】

【課題】

新規のデータ構造を導入して高速処理可能な、並列コンピュータのためのコンパイラを提供する。

【解決手段】

並列化指示文を含むソース・プログラム10をコンパイルするコンパイラ20は、ソース・プログラム中の並列化指示文を検出するステップと、並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップとを実行する。また、並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から当該並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップも実行する。

【選択図】　図３

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社